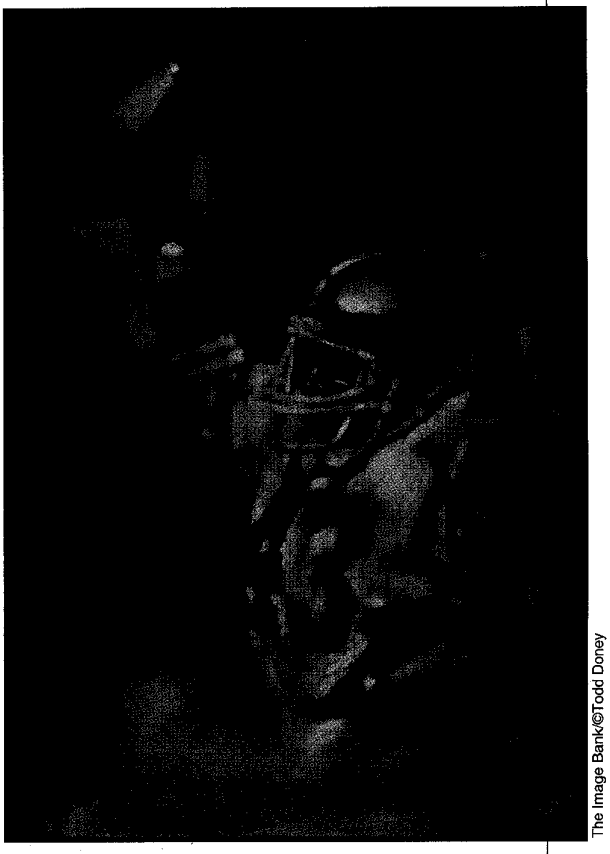
Neural network quarterbacking

How different training methods perform in calling the games

Michael C. Purucker

In the National Football League (NFL), teams that outperform their opponents in four categories usually are victorious. These statistics are yards gained, rushing yards gained, turnover margin, and time of possession. In fact, through the first eight weeks of the 1994 regular season, no team leading its opponent in all four categories had lost. In general, the more categories a team leads, the greater its chance of winning a game. Therefore, the relative strength of NFL teams can be established by comparing these four statistical categories.

In this article, several neural network strategies are tried to predict winners in NFL games. Binary, ternary, and continuous input vectors are used as inputs to appropriate networks: Hamming, Adaptive Resonance Theory (ART), Kohonen Self-Organizing Map (SOM), and Back-Propagation (BP). Predictions are presented, and the performance of each network is examined. Network results using supervised and unsupervised training methods are also compared.



The Image Bank/©Todd Doney

Background

Neural networks have an inherent advantage because of the lack of restrictions placed on the network. Freedom from these restrictions, which exist in statistical formulas due to algorithmic specification or the requirement of sequential input presentation, gives the network more flexibility and adaptability to changing inputs. Alternatively, if rigidity is desired, the network can be frozen at any time in its current arrangement.

A neural network has a certain number of simple processing elements called neurons. Each one is connected to other neurons by means of directed communication links. Each link has an associated weight. The weights collectively represent information being used by the network to solve a problem.

A neural network is characterized by its architecture, training, and activation function. Architecture describes the pattern of connections between neurons. Training indicates the method used to determine the weights associated with links. Typical activation functions are nonlinear, such as the sigmoid and hyperbolic tangent (tanh) functions. The activation or internal state of a neuron is determined by the application of the neuron's activation function to the weighted inputs, resulting in an output signal. Only one activation can exist at a given (discrete) time, but it can be sent to several neurons as an input (Fig. 1).

No one standard neural network architecture exists; users often try many variations before settling on a network type. Regardless of type, the number of input and output neurons must be chosen to fit the available data and desired results. Information to be processed by a neural network is called a pattern, and is presented in vector form. Input vectors are 1 x n arrays of signals that provide one value, discrete or continuous in form, for each of n input neurons. Thus, the size of the input layer depends on the size of the array. This in turn depends on the amount of information contained in these patterns. There must be enough input neurons to allow distinguishing dissimilar patterns while not separating those that belong together. However, the degree of precision necessary to group patterns together varies depending on the chosen application.

The presentation of an entire set of patterns to the network is called an epoch. Thus, the number of epochs is a measurement of the number of times a complete set of input vectors has been presented to the network. The group of patterns presented for training purposes is termed a training set. In most cases, training sets are presented in random order, varying with each epoch. The relative success or failure of a neural network often depends on the preparation of the input vectors.

0278-6648/96/$5.00 © 1996 IEEE

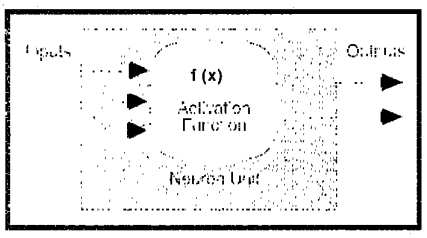


Fig. 1 An individual neuron.

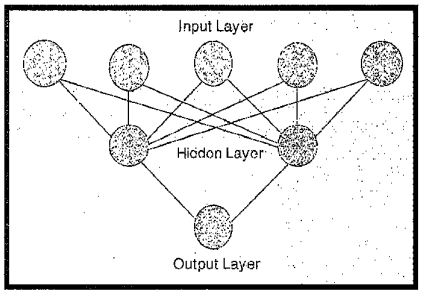


Fig. 2 Multilayer perceptron architecture. The above network has 5 input neurons, 2 hidden neurons, and 1 output neuron. All lines are weighted neuron links. Bias units are not shown in the figure.

Supervised training

Back-Propagation (BP) neural networks use a training method in which the difference between the actual network output and a supplied target output is back-propagated to adjust the network weights and reduce the error present. This error calculation and adjustment process is undertaken for each pattern. One prevalent neural network is a multilayer perceptron trained with the BP algorithm. A typical multilayer perceptron consists of three layers: input, hidden, and output layers. The input layer neurons receive their signals from an input vector. The hidden layer receives weighted inputs from the input layer neurons and processes these signals using an activation function. The hidden layer outputs are sent as weighted inputs to the output layer and processed. The outputs of the output layer neurons are the outputs of the neural network. One possible arrangement is presented in Fig. 2.

The hidden layer can range from multiple layers of several neurons each to no neurons at all. Applications involving large amounts of information or complex tasks of discrimination usually require more hidden neurons. The output layer can be as small as one neuron but is not limited to any particular number. Therefore, an almost infinite number of BP architectures can be constructed for use in applications.

Particular diligence must be applied when choosing hidden layer configurations for a multilayer perceptron. An architecture with too few hidden neurons results in a network that is unable to distinguish patterns; an architecture with too many hidden neurons can result in a network that is simply memorizing the training set patterns. It will be unable to make the generalizations required to correctly process the test data upon presentation.

The multilayer perceptron trained with the BP algorithm is the only network in this article that utilizes supervised training. BP networks can accept discrete or continuous input vectors. Network output targets inform the network which patterns indicate success and which patterns indicate failure based on actual prior results. During training, weights are updated to move the network closer to the given network output target in the following manner:

1. Each output unit compares its computed activation with its target value to determine the associated error for a particular pattern.

2. An error-dependent factor is calculated. This factor is propagated back to the previous layer of neurons. Later, it is used to update the weights leading to the output layer neurons.

3. Similarly, error factors are propagated recursively back through each layer of (hidden) neurons until the input layer is reached. Input neurons are not changed.

4. All weights are adjusted simultaneously at the end of each epoch based on the error factors at each layer.

In BP, the learning coefficient defines the rate of learning and the momentum value adds a tendency for weights to change in the direction they have been changing.

Unsupervised training

Unsupervised training attempts to cluster similar patterns together without using training data. In theory, the patterns are classified based on similarities between input vectors—there are no targets. Network weights are updated so that similar input vectors are assigned to the same output cluster.

Classification decisions are made by assigning input vectors to the cluster that features the most similar exemplar vector. An exemplar vector is the role model or representative of a class of vectors. It is used as a basis of comparison prior to admitting another vector to the class. Some networks rigidly maintain their exemplars, while others update exemplars to reflect the most recent addition. This is done either by making the new entry to a class the exemplar vector or through some variation of a weighted average of all of the vectors in a cluster.

Adaptive Resonance Theory (ART) networks begin the classification process by creating a cluster for the first input vector. A new class is created when an input vector exceeds a selected level of variance compared to the first cluster. Additional classes are created each time successive input vectors exceed a selected level of variance compared to the currently existing clusters. This level of variance is called the vigilance. The number of clusters to be formed affects the performance of the ART network. Like BP, ART networks can utilize discrete or continuous input vector values.

A Hamming network is a maximum likelihood classifier that can determine which of several exemplar vectors is most similar to an input vector. Input vectors are clustered based on the minimum Hamming distance between vectors in a class. Lippman states, "The Hamming distance is the number of bits in the input which do not match the corresponding exemplar bits." In the Hamming paradigm, exemplar vectors are stored in the initial weights. Upon presentation of an input vector, the output with the higher activation is considered the winner. The exemplar representing that output is chosen as most similar to the input. Hamming networks accept discrete (bipolar) input vector values.

In Fig. 3, "Maxnet" indicates a subnetwork that functions to select the neuron with the largest input. The Maxnet is an example of the "winner takes all" strategy, in which the neuron possessing the highest activation is rewarded with updates. All other neurons are reduced to zero values. This update strategy enhances the contrast present between different neuron activation levels.

Like the Hamming network and the Maxnet subnetwork, Kohonen Self-Organizing Map (SOM) networks use updating schemes based on competition. Kohonen SOM networks assume the existence of a topological structure among cluster units (output neurons). Thus, it is also classified as a topology-preserving map. The weight vector for a cluster serves as an exemplar for the input

patterns associated with that cluster.

Self-organization occurs by assigning the input vector to the cluster whose weight vector most closely matches the input pattern. (Typically, this is done by comparing the squares of the minimum Euclidean distance of competing cluster units.) Weights of the winning cluster unit and its neighboring units are then updated.

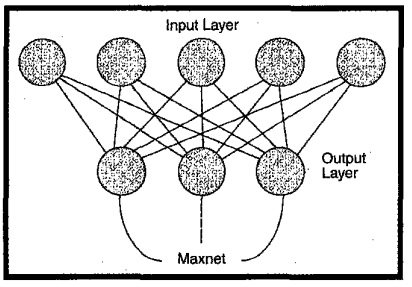


Fig. 3 Hamming network with 3 exemplars. There are 5 input neurons. The lines are weighted neuron links. (The Maxnet links are of the same weight.)

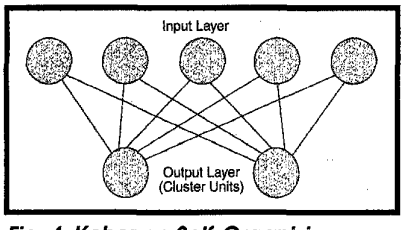


Fig. 4 Kohonen Self-Organizing Map. There are 5 input and 2 output neurons. The lines represent weighted neuron links.

The topological neighborhood radius decreases as the clustering process continues. Initial radius parameters are set by the user. Figure 4 shows one Kohonen SOM architecture. Both continuous and discrete input values are accepted by the input neurons of the Kohonen SOM.

Although the weights represent the information content of the SOM network, they are not multiplied to the signals sent from the input neurons to the cluster units. Weights can be assigned random values initially; however, if some insights are available with respect to a particular application, this information can be reflected through the appropriate choice of initial weights. Although initial map formation occurs quickly, many epochs may be necessary to reach final convergence during testing.

Methods

The ability to decipher trends and/or patterns in large amounts of data is an asset of neural networks. To utilize this attribute, results from previous NFL games were analyzed and processed; then these results were used to predict the results of the following week. For example, Week 15 games were predicted based on statistics from Weeks 12-14. The process of providing input to the network was performed as follows:

1) Statistics for each team were collected from box scores in the *Pittsburgh Post-Gazette* and the *Washington Post.*

2) The statistics were then encoded. For the continuous-input neural networks, aggregate statistics from the previous three games were processed to produce five input vector values for each team (all 3-game cumulative totals):

victories;

(yards gained—yards allowed)/100;

(rushing yards gained—allowed)/50;

turnover margin (fumbles, interceptions);

possession time margin in minutes/10.

To produce discrete input vectors, the continuous values were appropriately scaled: for binary networks, inputs represented positive values as 1 and negative values as 0; for ternary networks, inputs represented positive values as 1, values near zero as 0, and negative values as -1.

3) For the Back-Propagation method only, training input vectors were assigned targets: 1 for wins, -1 for losses. Instead of focusing on the matchups between opponents, each input set of 28 vectors represents the performance of individual teams over the previous three weeks. Network outputs thus reflect the relative strength of all 28 teams, producing a rank order or power rating. Game predictions are made by comparing the ranks of opposing teams for each scheduled game, with the higher-ranked team predicted to win.

Encoding considerations

The discrete-input Hamming network was trained using two different exemplar sets. One set attempted to separate winning teams from losing teams, while a three exemplar set attempted to separately cluster winning, losing, and marginal teams.

Discrete inputs also were presented to the Adaptive Resonance Theory network (ART, specifically ART1, Carpenter & Grossberg, 1987a). Like the Hamming network, ART1 was trained with the original five inputs only. Several combinations of vigilance level, exemplar classes, and input vector presentation order were tested. In each case only one epoch was required to cluster the data.

Continuous input vectors were used in the Kohonen Self-Organizing Map (SOM) network. The training length selected for the SOM networks was 10 epochs. This was found to produce sufficient separation and clustering of input vectors. Also, a 10x10 neuron grid and coordinate output layer with interpolation was used to implement the SOM network and produce meaningful outputs. Sufficient clustering occurred for each set of inputs. This resulted in at least two groupings—strong and weak teams—and often many intermediate levels.

Back-Propagation (BP) networks also were presented with continuous input vectors. Significantly, BP was the only network that received supervised training. To train the BP network, targeted input vectors were produced from "exemplar" teams—the four teams with the best NFL records and the four teams with the worst NFL records. San Francisco, Dallas, Pittsburgh, and Cleveland, with a combined record of 41-11 through their collective first 13 games, provided input vectors representative of victory profiles through their statistics in Weeks 11-14. To further improve the victory model, information from the losses by Dallas and Cleveland in Week 11 and again by Cleveland in Week 13 were omitted from the training, for a total of 13 victory input vectors.

Similar analysis of statistics from Houston, Cincinnati, Washington, and the Los Angeles Rams, a combined 9-43 in their first 13 games, resulted in 14 defeat-profile input vectors. Thus, a total of 27 input vectors were derived from the best and worst teams in the NFL to provide guidance for the BP network. In a sense, the training noise was minimized by developing the training file in this manner. Given the nature of the NFL, reducing noise during training is imperative to network success.

The BP network was trained with targets, 1 indicating victory and -1 indicating defeat. Training lengths of 10, 50, and 100 epochs were used. Sample training and testing files are presented in Fig. 5. Inputs for each vector are arranged in the order stated previously with an additional, sixth element added to the training file vectors to represent the targeted output.

A learning coefficient of 0.5 and a momentum value of 0.4 were used in the BP updating schemes. A cumulative update strategy was used. The weight changes were accumulated over an epoch before being applied to the network. Various fully-interconnected network architecture configurations including zero, one, and two hidden layers were tested for effectiveness and efficiency.

The hyperbolic tangent (tanh) activation function was used in each network. Biases were supplied to neurons as required by paradigms. Every effort was made to produce a consistent and impartial environment for each network to allow for true comparisons between networks based on performance.

Unsupervised training results

No meaningful difference existed between the outputs of the Hamming network with three exemplar classes and the two exemplar network. In each case, classification broke down into groups of 10 and 18 teams, each of which was 50% successful (5 of 10, and 9 of 18) in Week 15 of the NFL season. Essentially, the Hamming network classified as if by chance. It was unable to distinguish teams based on the input vectors. In fact, it was impossible to tell which teams it selected to win and which it predicted to lose.

However, the Hamming network was not expected to fare well. Its main role is to classify binary patterns and identify the randomness of noise in data. It is not designed to function well with input vectors that have large Hamming distances substantially different from the exemplars. Also, even if the statistics are discretized, they still retain their continuous nature. As such, they cannot be accurately represented discretely.

The Adaptive Resonance Theory (ART1) network also struggled with Week 15 predictions. In its best performance, it predicted seven of the winners while selecting four losing teams and not predicting three matchups, for a success rate of only 50%. This average to poor performance was again a function of the input data. It was obvious that the network would struggle simply by examining the input vectors prior to implementation of the ART1 network. The five encoded statistics were reduced to a 1 or 0. This makes it difficult for the network to learn the trend differences between winning teams and losing teams.

The ART1 results were consistent across many vigilance levels. Input presentation order held no consequence to the network. Performance only decreased substantially upon restricting the number of output classes to two. Evidently, losing teams displayed greater variance than winning teams. Thus, they were clustered into multiple classes. Restricting the output classes to two appeared to force teams otherwise classified in the winning team cluster to join the losing team cluster. Although this result did not facilitate the prediction of games, it was similar to the real-life perception of the NFL: when dividing teams into only two groups, a select few are considered winners, and the vast majority are grouped together as losers.

In *Natural and Artificial Intelligence* (1992), deCallatay alludes to the inability of ART to obtain accurate solutions by combining inferences. He describes a scenario similar to using statistical inferences to predict game outcomes: "Grossberg underlines that a main problem is to maintain stability in plasticity... ...I do not see how ART could solve problems by combining inferences."

"Maintaining stability in plasticity" refers to the difficulty in developing an ART network that simultaneously possesses sufficient flexibility and robustness. (That is to respond to input changes through network modification while maintaining its integrity based on previous information.)

In fact, all neural networks struggle to achieve this balance. This task is difficult enough without contending with occasionally misleading statistics. Teams have a disturbing tendency to play to the level of their opponents and/or to be affected by emotional factors. This volatility is attributable largely to human nature.

Neural networks in general—and ART in particular—are affected by this volatility, although not as much as purely mathematical methods. Since knowledge is decentralized in networks, volatile effects are lessened. In addition, it is in *combining inferences* where neural networks can exceed the performance of statistical methods. However, finding the best combination of input data presentation and network architecture remains a formidable challenge.

From their performance in Week 15, it appears discrete-input networks cannot adequately predict NFL games. This mediocre performance is not surprising. Much information is lost when statistics are discretized. Despite the limited information content of discrete inputs, better results might still be achieved by using a paradigm that applies supervised training methods.

In any case, better results were obtained with continuous-input networks. The Kohonen Self-Organizing Map (SOM) network performed reasonably well, producing a record of 8-6. Although 57.1% is not an exceptional performance, it does compare closely with many football "experts." Certainly, it was the best performance by a network

bptrain.nna	*contest.nna*
!WINNING TEAMS	!SF,Dallas,Pitt,Clev
!Pitt Weeks 11-14	3 5 5 4 5
2 0 2 -2 0 1	3 0 3 3 0
2 -1 0 0 -1 1	3 3 5 7 3
3 0 1 4 0 1	1 1 0 -8 -1
3 1 3 5 0 1	!SD,Miami,Minn,Det
!Clev Weeks 11 & 13	1 -3 0 3 -2
2 -1 0 3 -2 1	1 -1 -1 -6 -2
2 0 0 0 -1 1	1 2 1 -3 0
!SF Weeks 11-14	2 -1 0 5 -3
3 4 4 7 1 1	!Chi,GB,Buf,NE
3 3 4 4 1 1	2 1 2 -4 3
3 1 3 3 2 1	0 -3 -6 -1 -2
3 2 4 3 3 1	2 1 3 0 2
!Dallas Weeks 12-14	3 3 1 1 4
2 4 1 -3 3 1	!KC,Raid,Den,Phil
2 3 2 2 2 1	1 1 1 -1
2 2 2 0 1 1	2 2 1 -5 0
	3 0 0 2 3
!LOSING TEAMS	0 -1 0 0 -2
!Hou Weeks 11-14	!Az,Atl,Indy,Jets
0 -1 -1 -2 0 -1	2 1 3 8 2
0 -1 1 2 1 -1	1 -2 -3 -3 -1
0 -1 2 1 0 -1	2 -1 2 2 -2
0 -2 1 3 0 -1	1 -2 -1 6 0
!Cinc Weeks 12-14	!NYG,NO,Sea,Hou
2 2 -4 -2 -2 -1	3 0 0 4 2
2 4 -2 -5 0 -1	1 -4 -3 5 -1
1 3 -1 -8 -1 -1	2 -1 -1 0 -1
!Wash Weeks 12-14	0 -3 -3 -4 -4
1 -1 0 -1 -1 -1	!Cinc,Wash,TB,Ram
0 0 -2 -8 -1 -1	0 1 0 -9 -2
0 -1 -2 -5 -2 -1	0 -1 -5 -5 -4
!Rams Weeks 11-14	2 2 3 3 3
2 0 1 -3 0 -1	0 0 -3 -6 -2
1 0 0 -4 -2 -1	
1 0 0 -1 -4 -1	
0 0 -4 -3 -3 -1	

Fig. 5 Training and Testing Files. The BP training file, bptrain.nna, is representative of "noiseless" input vectors. The test file, contest.nna, has no targets, and is also used as the input file for the Kohonen SOM network. The order of inputs in the figure follows that outlined in Methods.

Table 1: Back-propagation comparison training length vs. output

Team	Result	vs.	100 epochs	50 epochs	10 epochs
PITT	W 14-3	PHIL	1.222	1.172	0.789
SF	W 38-15	SD	1.209	1.143	0.679
NYG	W 27-20	CINC	1.182	1.120	0.773
DEN	L 13-23	RAID	1.174	1.107	0.738
DAL	L 14-19	CLEV	1.168	1.089	0.798
NE	W 28-13	INDY	1.148	1.063	0.644
AZ	W 17-15	WASH	1.117	0.989	0.461
TB	W 24-14	RAMS	0.976	0.803	0.310
BUF	L 17-21	MINN	0.805	0.626	0.288
DET	W 18-7	JETS	0.667	0.575	0.459
INDY		NE	0.661	0.532	0.419
CHI	L 3-40	GB	0.548	0.399	0.173
SEA	W 16-14	HOU	0.394	0.355	0.323
JETS		DET	0.266	0.178	-0.018
NO	W 29-20	ATL	0.062	0.066	0.014
RAID		GB	0.548	0.399	0.173
SD		SF	-0.134	-0.139	-0.117
KC	L 28-45	MIA	-0.504	-0.458	-0.214
MINN		BUF	-0.784	-0.698	-0.350
ATL		NO	-0.832	-0.672	-0.254
MIA		KC	-1.000	-0.875	-0.321
CLEV		DAL	-1.062	-0.985	-0.429
PHIL		PITT	-1.088	-1.011	-0.629
GB		CHI	-1.163	-1.071	-0.661
HOU		SEA	-1.199	-1.137	-0.665
RAMS		TB	-1.215	-1.168	-0.787
CINC		NYG	-1.223	-1.189	-0.823
WASH		AZ	-1.226	-1.183	-0.758

Teams in blue were selected by the network to win, by virtue of being ranked above their opponent. The result column indicates whether the selected team won (W) or lost (L) and the final score. This network predicted 9 out of 14 games correctly (64.3 percent) in Week 15.

using unsupervised training methods, probably due to its continuous inputs.

Supervised training results

The multilayer perceptron architecture shown in Fig. 2 was used as the initial configuration for Back-Propagation (BP) network testing. Since predictions were similar for various hidden layer architectures, the simple five-input, one-output, no hidden layer architecture was chosen. Continuous-input vectors were used initially with BP.

Output values for the training lengths of 10, 50, and 100 epochs are presented in Table 1. Game predictions are consistent at each training length; but, some slight changes occur in overall team rankings as training length increases. Sufficient separation was found to occur using a training length of 50 epochs.

To predict game outcomes, the outputs of the competing teams were compared. The team with the higher output was predicted to win. With only the five standard input data values, nine out of 14 games (64.3%) played were predicted correctly using continuous inputs during Week 15. Using discrete inputs, BP successfully predicted eight out of 14 games (57.1%).

Although surprisingly similar in performance, discrete-input networks are at a significant disadvantage. Perhaps this disadvantage is best illustrated by example. An input vector of [3 4 7 5 3] would be converted into discrete form as [1 1 1 1 1]. However, an input vector of [2 1 1 1 1] also would be converted to [1 1 1 1 1].

Combining all network results, BP has demonstrated both objective and subjective advantages over the other networks. Objectively, BP exceeds their performance. Subjectively, BP ranked the teams in an order consistent with the relative strengths of teams as shown during the 1994 season; that is, in agreement with the opinions of football experts.

An interesting effect from some hidden layer configurations was the clustering of teams around similar numerical values. For example, using two hidden layers (three neurons) and continuous inputs, nine teams had values within 0.001 of each other. In instances where these teams are playing each other, no prediction can be offered on the game.

One possible explanation is that the network is overfitting the data, and is essentially memorizing the training set. Thus, when the test set is supplied, it has not learned to distinguish between patterns. From a different perspective, however, this may be a desirable outcome; the network is concluding that predictions should be avoided since the teams are too evenly matched.

Impact of training

Training the BP network with input vectors from exemplar teams is arguably its most significant advantage over the other networks. Visual inspection of the bptrain.nna file in Fig. 5 immediately shows that the winning teams (Pitt, Clev, SF, Dallas) contain a majority of positive input values. The losing teams (Hou, Cinc, Wash, Rams), however, have predominantly negative input values. Excluding the first input value, which represents victories in the previous three games, the winning teams have negative values for less than 15% of their inputs. Losing teams have positive values for less than 20% of their inputs.

A neural network is not necessary to observe that good teams possess positive input patterns, and bad teams have negative input patterns. However, supervised training allows for more sophisticated network learning to occur. Based on the values of specific inputs in the exemplar team patterns, the network learns which encoded statistics are more indicative of victory. Then upon presentation of the test set, the network produces output values that reflect how closely the test inputs match the profiles of winning or losing exemplar teams.

Training set vectors containing negative input values on winning exemplar teams and positive values on losing exemplar teams will result in altering the weights of the network. Thus, the network learns to de-emphasize the importance of the input with the abnormal value.

Similarly, when using continuous inputs, a large value for an input will place greater emphasis on the statistic it represents. More importantly, the statistics that are least or most significant can change over time. The network will automatically adjust to these changes since it relies solely on the current training set. Clearly, supervised training depends heavily on the encoding paradigm and the statistics represented by that scheme.

Enhanced input

After initial testing, two additional inputs were encoded in an attempt to

increase the information content of the input vectors. Inspired by expert systems approaches, one input would attempt to convey "the knowledge of human experts" as defined by Van Horn. The other input would encode schedule difficulty to reflect the caliber of opposition, thereby providing context for the other statistics compiled by teams.

Football experts use much more than just statistics to predict the results of games. Although a 64.3% success rate compares favorably to many football experts (*Pittsburgh Post-Gazette* football writer Gerry Dulac had predicted <64% correct through 15 weeks of the 1994 season), some methods to improve the success rate must be considered. Winning has certain elements that cannot be captured in numbers. In attempting to include some of these intangible values, the additional input should:

1) be applicable to all teams as a consistent indicator of the relative strength of teams based on performance standards;

2) take into consideration factors such as home field advantage, playing surface, expected weather conditions, team and player matchups, and others known to impact outcomes;

3) be easily accessible;

4) be accurate and simple to encode.

One existing item fits all of the above listed criteria: the Las Vegas line. Every NFL game is addressed by the line, and a point spread is presented favoring one team over another. Only one strong caution must be applied to use of the betting line: it is designed to produce an even amount of public betting on both teams in a game, *not* to establish the better team or the team expected to win.

At first glance, including a schedule difficulty factor would appear to augment network performance. Encoding is simple, accomplished using the combined overall records of opponents from the previous three weeks. Certainly, this information is easily accessible. It applies to all teams, indicates relative strength to some extent, and could be expanded to reflect performance on specific playing surfaces and home/away results. More importantly, a schedule difficulty factor could serve the purpose of qualifying the other statistics included in the input vector.

Preliminary testing, however, suggested otherwise. Multiple attempts to encode an appropriate schedule difficulty factor produced negative results. For each of several encodings, one or two of the previously correct predictions were changed; however, no incorrect predictions were reversed. In addition, the overall power ranking of teams produced by the network disagreed with the perception of football experts to a greater extent.

Upon closer inspection, the reason the schedule difficulty factor introduced inaccuracies is that team records may be inflated due to playing weak opponents. Also, records do not take into account the margin of victory or the quality of the victory, i.e. "winning ugly" vs. dominating an opponent. Perhaps encoding the overall records of opposing teams introduces contradictions within certain input vectors. In any case, network performance was not enhanced, so the schedule difficulty factor was eliminated.

Adding more expertise

The Las Vegas line was encoded as a parameter to the team favored by the line. Methods were used consistent with the prior encoding of input data. The Las Vegas line was encoded as a positive parameter to the team favored by the line. New training and test files were created, each including the betting line input. The same procedure was utilized to train the network, featuring exemplar teams and targets.

The additional information enabled the BP network to improve its Week 15 performance by one game (10 out of 14 games or 71.4%). Without it, Kansas City had been chosen to defeat Miami. (Miami won 45-28.) In addition, it made the BP decision closer on one game that was chosen incorrectly. The choice of one win was also made closer, but the network maintained its correct choice. All other game predictions were relatively unchanged.

Like the BP network, the Kohonen SOM network improved its performance by one game with this additional information. The network now chose nine correct out of 14 (64.3%) for Week 15. Unlike BP, however, the Kohonen SOM network was overwhelmed by the new information, choosing the favored team to win in all 14 games (see Table 2). This overreaction is not helpful when predicting games. By comparison, the BP network chose only nine of the favored teams to win, an increase of one (Miami), with the additional information. These results demonstrate the robustness of BP in comparison to the Kohonen SOM network, at least for Week 15.

Table 2: Network predictions agreement with Las Vegas line

Week 15	
Back-Propagation	8 of 14
Back-Propagation with LV line	9 of 14
Kohonen SOM	9 of 14
Kohonen SOM with LV line	14 of 14
Week 16	
Back-Propagation	9 of 14
Back-Propagation with LV line	12 of 14
Kohonen SOM	7 of 14
Kohonen SOM with LV line	9 of 14

Week 16 predictions

The results from Week 15 demonstrated the superiority of the continuous-input BP and Kohonen SOM networks for prediction. To further analyze these two networks, to examine the effect of the Las Vegas line as an additional input, and to enhance the legitimacy of the neural network prediction scheme, the NFL games of Week 16 were predicted. Using the same protocols for encoding, training and testing, the continuous-input BP and Kohonen SOM network predictions for Week 16 of the NFL are presented in Table 3.

One minor change was made in the encoding method to better represent the cumulative statistics of the previous three games. For Week 16 predictions, encoded values were rounded off only after all three weeks of statistics had been added together. For Week 15 games, encoded values derived from statistics from each of the three weeks had been rounded off to the nearest integer or to the nearest 0.5 prior to summation.

Although unlikely to produce a major impact, this strategy more accurately represents the NFL statistics in encoded form. A high level of agreement exists between the four sets of neural network predictions listed in Table 3. Consensus agreement exists for five games. Also, with the line included as input, the Kohonen SOM agrees with BP on nine of its 14 predictions, with one tie as well.

As illustrated in Table 2, a different week of games, accompanied by a new set of training data, produced different predictive pattern characteristics in the two networks. In Week 16, BP shows a greater tendency to agree with the team

favored by the Las Vegas line, choosing those teams in nine games with its original five inputs, and agreeing in 12 of 14 games with the additional information. In contrast, the Kohonen SOM network selects the favored team in seven games initially, and in nine out of 14 games with the additional information

The enhanced-input Kohonen SOM network gains credibility by disagreeing with the Las Vegas line in some Week 16 games. This is in contrast to the complete agreement it exhibited in Week 15. However, it remains more volatile than BP. The Kohonen SOM network changed half of its predictions once exposed to the line, whereas BP changed in only two of the 14 games.

Results

BP successfully predicted 11 of 14 games (78.6%) in Week 16 versus Kohonen SOM which selected only 8 of 14 games correctly. (However, with the line, each network chose 11 of 14 games correctly.) The better performance of BP is not a coincidence; the target vectors provide more information to BP than is present in the Kohonen SOM network. This makes the BP a more intelligent model.

In addition, all three games predicted incorrectly on BP's part were close. For the six close games (decided by less than 10 points) in Week 16, BP selected 50% correctly. In the eight games decided by 10 points or more, BP was 100% successful.

Conclusions

The following conclusions can be drawn from the research on the use of various neural network algorithms to predict football games:

• Input information content and supervision of training are essential to network performance. Information content is principally determined by properties of the input vector. Network knowledge is primarily a function of the degree of supervision during training. However, a multiplicative effect does exist between input information content and training supervision. This is because increased information allows for better training.

• Predictions for Week 16 exceeded the performance of many football experts. However, these results should be tempered with the reality that a high level of noise and randomness exists in this system. Any method for predicting events involving hundreds of human beings cannot be completely captured by an encoding scheme.

• Further improvements in performance may be achieved through the fine tuning the encoding process, network architecture and training methods over the course of a full NFL season.

Note that football common sense must be applied to network predictions. There are four immediate sources of error. Early season games have no statistical track record for encoding the network. Late season games may be affected by teams resting players or preparing for the future instead of focusing solely on winning. Finally, weather and injuries can play havoc with game results. Week 16 results could have been the same as in Week 15 or possibly worse because football games are so volatile.

Table 3: NFL Game Predictions Week 16, December 17-19, 1994

All Networks
San Francisco def. Denver
Pittsburgh def. Cleveland
Tampa Bay vs. Washington
N.Y. Giants def. Philadelphia
Kansas City def. Houston

Back-propagation
Miami def. Indianapolis
Arizona def. Cincinnati
Green Bay def. Atlanta
Chicago def. L.A. Rams
Detroit def. Minnesota
San Diego def. N.Y Jets
New Orleans def. Dallas
Seattle def. L.A. Raiders
New England def. Buffalo

Back-propagation with LV line input
Miami def. Indianapolis
Arizona def. Cincinnati
Green Bay def. Atlanta
Chicago def. L.A. Rams
Detroit def. Minnesota
San Diego def. N.Y Jets
New Orleans def. Dallas
L.A. Raiders def. Seattle
Buffalo def. New England

Kohonen SOM
Miami def. Indianapolis
Cincinnati def. Arizona
Atlanta def. Green Bay
L.A. Rams def. Chicago
Minnesota def. Detroit
TIE San Diego/N.Y Jets
Dallas def. New Orleans
L.A. Raiders def. Seattle
New England def. Buffalo

Kohonen SOM with LV line input
Indianapolis def. Miami
Arizona def. Cincinnati
Green Bay def. Atlanta
Chicago def. L.A. Rams
Detroit def. Minnesota
N.Y Jets def. San Diego
Dallas def. New Orleans
Seattle def. L.A. Raiders
TIE New England/Buffalo

The actual results from Week 16

San Francisco 42	Denver 19	Green Bay 21	Atlanta 17
Pittsburgh 17	Cleveland 7	Chicago 27	L.A. Rams 13
Tampa Bay 17	Washington 14	Detroit 41	Minnesota 19
N. Y. Giants 16	Philadelphia 13	San Diego 21	N.Y. Jets 6
Kansas City 31	Houston 9	Dallas 24	New Orleans 16
Indianapolis 10	Miami 6	L.A. Raiders 17	Seattle 16
Arizona 28	Cincinnati 7	New England 41	Buffalo 17

Acknowledgments

The neural networks used in this study were developed by NeuralWorks Inc. and were accessed through their 1994 NeuralWorks Explorer PC software package.

Thanks to Dr. Alice Smith, instructor, Neural Networks & Applications, at the University of Pittsburgh, for assistance in the revision of the original paper.

Read more about it

• Graphic, *Monday Night Football*, ABC Television Network, October 31, 1994.

• L. Fausett, *Fundamentals of Neural Networks*. Prentice-Hall: Englewood Cliffs, New Jersey, 1994.

• R.P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE-ASSP*, April 1987.

• Box scores and Las Vegas betting line, *Pittsburgh Post-Gazette*, Sept.-Dec., 1994.

• Box scores, *Washington Post*, Sept.-Dec., 1994.

• M. Van Horn, *Understanding Expert Systems*. The Waite Group, Bantam: New York, 1986.

• A.M. deCallatay, *Natural and Artificial Intelligence: Misconceptions about Brains and Neural Networks*. North-Holland: New York, 1992.

• Simon Haykin, *Neural Networks: A Comprehensive Foundation*, IEEE Press, 1994

About the author

Michael C. Purucker is a bioengineering graduate student at the University of Pittsburgh. He is funded through the Department of Rehabilitation Science and Technology for research under the supervision of Charles J. Robinson, D. Sc. His current research focuses on the characterization and control of a sliding linear investigative platform for analyzing lower-limb stability (SLIP-FALLS). This device will be used initially for studies on the psychophysics of balance and predictors of falling.